Further Application of Confidence Limits to Quantile Measures for the Lognormal Distribution using the MATLAB Program

Introduction

In the prior discussion as posted on the Petrocenter website, mean and confidence intervals (CI's) for quantile measures were developed using analytical equations and simulation. In this note, further applications of CI's are demonstrated using built-in functions in the MATLAB program. MATLAB provides a code that resembles the underlying mathematics in a more concise and simple manner than C++ or Visual Basic. The implementation of vector mathematics within MATLAB simplify the code so it can be readily understood.

The equations developed are general to any sample size and parent distribution. However, the focus of this and prior discussion is the estimators representing the distribution's tail end values from less likely events using order statistics for sample size of less than 300. This particular focus comes from real world problems, where repetitive sampling may be infeasible, and decisions must be made from the available data.

It is considered that a measure of the robustness or fragility of these sample estimates can be inferred by their theoretical CI's. The prior discussion developed CI's when percentile measures were used for values from exponential and Pareto I distribution. In this application, CI's of order statistics are developed for the log normal distribution.

Order Statistic Sampling Distribution and relationship to Percentiles

As derived in the prior note, the sampling distribution, g(x), of the X_k order statistic of an i.i.d. random sample size, *n*, taken from a parent population with probability density function (pdf) of *f(x)* is

$$g(x) = \beta(F(x)|a,b) \cdot f(x) \tag{1}$$

where $\beta(\cdot)$ is the beta probability distribution with parameters *a* = *k* and *b* = *n* – *k* + *1.* Derivation of equation 1 is given in the prior note.

Consider an order set of data, with x_1 as the lowest and x_n as the highest. V_p is defined as the value of the p^{th} percentile, where p is in the range of 0 to 1. In one method of calculating percentiles, the percentile estimate, V_p is the order statistic evaluated as *np* rounded up to the next whole integer. For example, if $np = 4.1$, then $V_p = x_5$. Therefore, the relationship of percentile and order statistics is defined as

$$k = \lceil np \rceil \tag{2}$$

$$V_p = x_k$$

The prior note provides alternative means of calculating percentiles and identifies the differences in these methods.

Confidence intervals of percentile measures can be directly calculated. The inverse cumulative function of g(x), given in the prior note is

$$G^{-1}(\rho) = F^{-1}[B^{-1}(\rho|a,b)]. \tag{3}$$

Equation (3) is used to calculate confidence limits. For example , a high confidence limit, CL(0.95) of the k^th ranked order statistic $CL(0.95)$ equals $G^{-1}(0.95)$. It is noted that $G^{-1}(0.50)$ is the median value of the sample. It is apparent that confidence intervals are a function of the parent distribution, its parameters, the given probability level, ρ, and the sample size and rank of the order statistic, as required to calculate *a* and *b*.

The expected value of the order statistic can be calculated as

$$E[X_k] = \int_{-\infty}^{\infty} x g(x) dx \qquad (4)$$

with g(x) defined in Equation 1.

Lognormal distribution

The log normal probability density distribution, with parameters μ and σ is given as:

$$f_X(x;\mu,\sigma) = \frac{1}{x\sigma\sqrt{2\pi}} e^{-\frac{(\ln x-\mu)^2}{2\sigma^2}}, \quad x > 0 \qquad (5)$$

where μ and σ are the mean and standard deviation of the random variable's natural logarithm (by definition, the logarithm of the random variable is normally distributed). The cumulative probability distribution is given as:

$$F_X(x;\mu,\sigma) = \frac{1}{2}\mathrm{erfc}\left[-\frac{\ln x-\mu}{\sigma\sqrt{2}}\right] = \Phi\left(\frac{\ln x-\mu}{\sigma}\right), \qquad (6)$$

The mean, variance and standard deviation of the random variable log normal distribution are functions of the μ and σ parameters:

$$\begin{aligned} \mathrm{E}[X] &= e^{\mu+\frac{1}{2}\sigma^2}, \\ \mathrm{Var}[X] &= (e^{\sigma^2}-1)e^{2\mu+\sigma^2} \\ \mathrm{s.d}[X] &= \sqrt{\mathrm{Var}[X]} = e^{\mu+\frac{1}{2}\sigma^2}\sqrt{e^{\sigma^2}-1} \\ \mu &= \ln(\mathrm{E}[X]) - \frac{1}{2}\ln\left(1+\frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2}\right), \\ \sigma^2 &= \ln\left(1+\frac{\mathrm{Var}[X]}{\mathrm{E}[X]^2}\right). \end{aligned} \qquad (7)$$

An excellent online discussion of the lognormal distribution can be found at the following link:

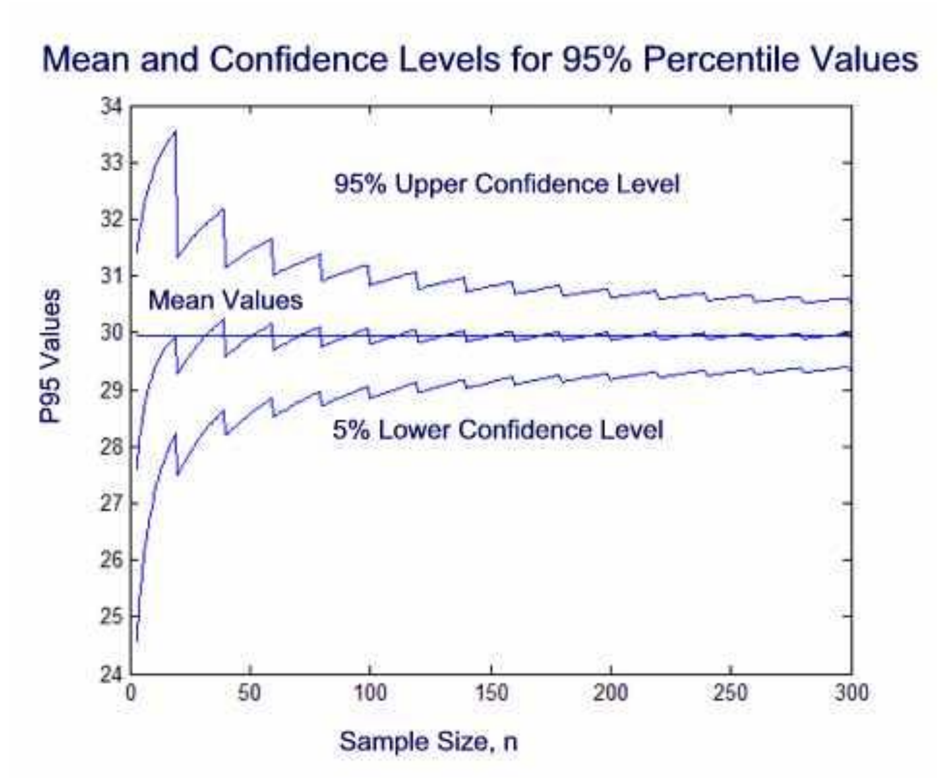
http://en.wikipedia.org/wiki/Log-normal_distribution

Example

The distribution parameters were calculated to match prior examples, with a theoretical 95% percentile value of 29.95. This value results for distribution parameters of $\mu = 3.235$ and $\sigma = 0.1$. The $E[X]$ and $Var[X]$ values are 25.41 and 6.55, respectively. The 95% percentile values (P_{95}) were estimated using order statistics as given in above (equation #2) using the built in functions available in the MATLAB program. The mean values of P_{95} were calculated using numerical integration. Confidence levels of 5% and 95% were used for this example, consistent with prior examples given in the paper and associated note.

As shown below, the mean values of P_{95} is underestimated for small sample sizes, consistent with prior discussion of the conservative nature of estimation using the method in Equation #2. Also, the confidence levels are generally balanced above and below the mean P_{95} value.

The mean value and confidence levels were also calculated using Monte-Carlo simulation. There was close agreement between numerical and simulation results.



Using MATLAB Built-in Functions to determine $E[X_k], CL(0.95), CL(0.05)$

Both numerical and simulation methods can be used to identify mean and confidence levels of the sampling distribution. For a given *n* value, the high and low limits are calculated for a 95% percentile limit as follows

```
k =ceil(n.*0.95);
a = k;
b = n-k+1;
xl = betainv(0.05,a,b);
x_low(i)  = logninv(xl ,mu,sigma);
xh  = betainv(0.95,a,b);
x_hi(i) = logninv(xh, mu, sigma);
```

with **ceil** = ceiling function, **betainv** = $B^{-1}(\cdot)$, **logninv** = inverse of the log-normal distribution. Subscript i is incremented each time a new value of *n* is calculated and used to build an array of results for plotting.

Calculation of the mean of the log-normal distribution is calculated by a quadrature (**quad** function) method

```
xmu(i) = the_integral(a,b, mu, sigma);
```

where the_integral function is

```
function xmu =the_integral(a,b,mu,sigma)
lower = logninv(0.001,mu,sigma);
upper = logninv(0.999,mu,sigma);
```

```
xmu = quad(@equ,lower, upper);
 function xmu = equ(x)
  y1 = lognpdf(x,mu,sigma);
  y2 = betapdf(logncdf(x,mu,sigma),a,b);
  xmu = x.* y1.*y2;
 end
end
```

The integration limits have been hard coded as inverse log normal values evaluated at 0.001 and 0.999 for convenience. The **quad** function will integrate a defined single argument function with no passed parameters.   To circumvent this restriction, a second function (@equ) is nested within the called function as all argument values to the called function are public. There are other means of circumventing the restriction noted in the help pages of MATLAB.

Simulation of sampling from a log-normal distribution uses the built-in random number generator to generate an array of m rows and n columns.   A listing of the simulation program is given in the Appendix.

References

1. Chapra, S., *Applied Numerical Methods with MATLAB for Engineering and Scientists*, McGraw Hill, 2005.

2. DeGroot, M. H, and Schervish, M., Probability and Statistics, Third Edition, Addison-Wesley, 2002, page 277- 279 (log normal distribution derivations).

Appendix: MATLAB Monte-Carlo Simulation using Lognormal Distribution

```
clear;
size = 10;
mu = 3.235;
var = 0.1;
p = 95; nsim = 1e6;
x = logninv(rand(size,nsim),mu,var)
z = prctile(x,p);
a= sort(z);
fprintf(1,' *** Case sample size %g \n',size);
fprintf(1, ' number of simulations %g \n',nsim);
ll = round(0.05*nsim);
ul = round(0.95*nsim);
al = a(ll)
au = a(ul)
am = mean(a)
% --- method 1
k = ceil(size*p/100);
x = sort(x);
b = x(k,:);
b = sort(b);
bl = b(ll)
bu = b(ul)
bm = mean(b)
% --- method 3
n1p = (size-1).*p/100;
k = ceil(n1p);
d = k - n1p;
v = d.*x(k,:) + (1-d).*x(k+1,:);
v = sort(v);
cl = v(ll)
cu = v(ul)
cm = mean(v)
```